Hands-On
JNI Development on Mac OS X

Session 631

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

Hands-On
JNI Development on Mac OS X

Matt Drance
Java DTS

http://developer.apple.com/wwdc2005/

What Is the JNI?

- Java Native Interface
- Integrate pure Java and native (C/C++/ObjC) code
- Useful for many developers
  - Porting native codebase to Java
  - Leveraging legacy/open libraries
  - Porting Java codebase to Cocoa/CoreFoundation
  - Provide platform features absent from Java

What You'll Learn

- How to build a JNI Application in Xcode
- Java/native GUI Integration
  - CocoaComponent
  - AWT Native Interface (JAWT)
  - Cocoa vs. Java event threads
- Communication between Java and native objects
- Invocation of a JVM from C/ObjC apps

Building JNI Projects With Xcode

http://developer.apple.com/wwdc2005/

Xcode and JNI Development

- Xcode is the only IDE that builds Java and native sources in one project
  - Java compilation
  - Native compilation/linking
  - Application bundling
- Start-to-finish JNI application with one Build button!

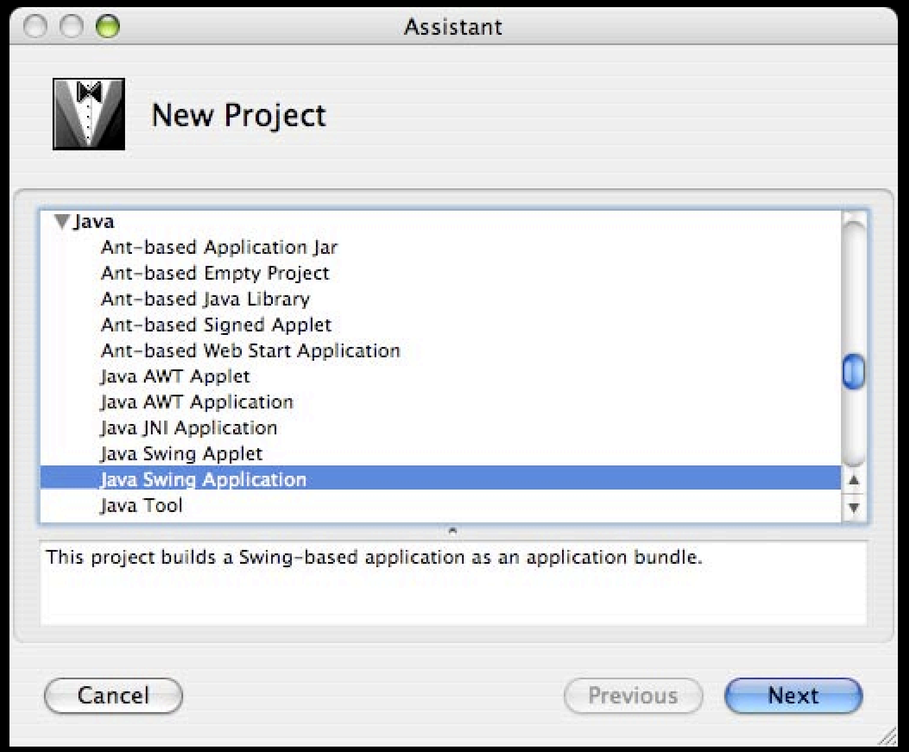
Creating a JNI Project in Xcode

- Java Application target
- Header target
- Native target(s)
- Dependencies
- Copy files phase

Demo
JNI Projects in Xcode

JNI Projects in Xcode

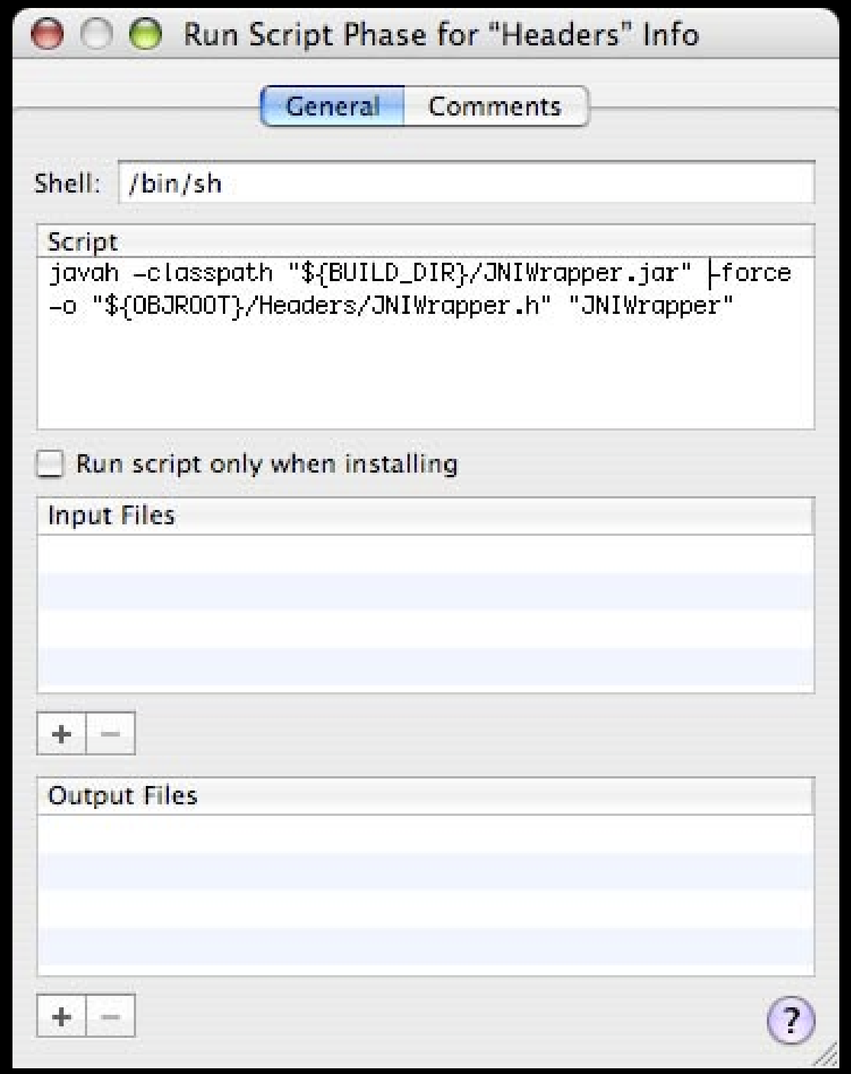
Java Application Target



- All Java classes belong in this target
- Set all properties, names as in a regular Java project

JNI Projects in Xcode

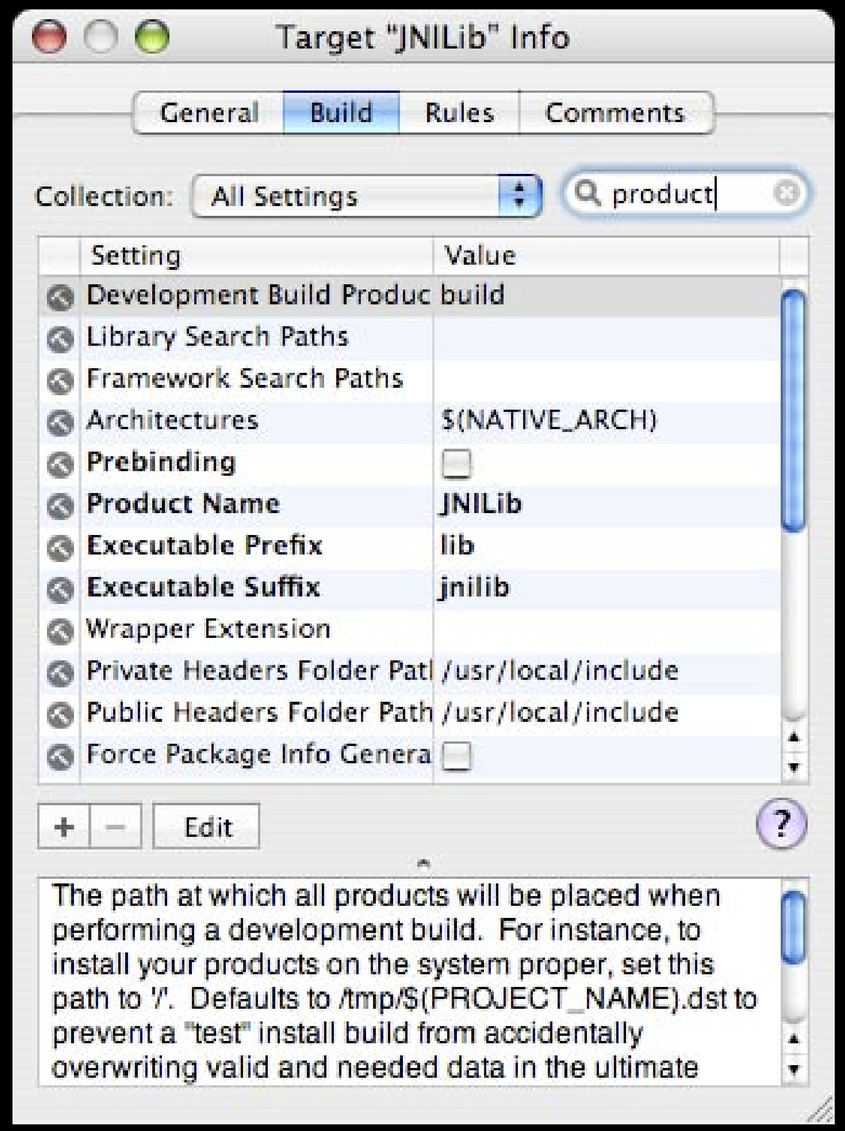
Native Header Target



- Isolate Native calls to minimal number of classes
- All classes with native methods go into this target
- No Dependencies
- Shell Script phase using javah

JNI Projects in Xcode

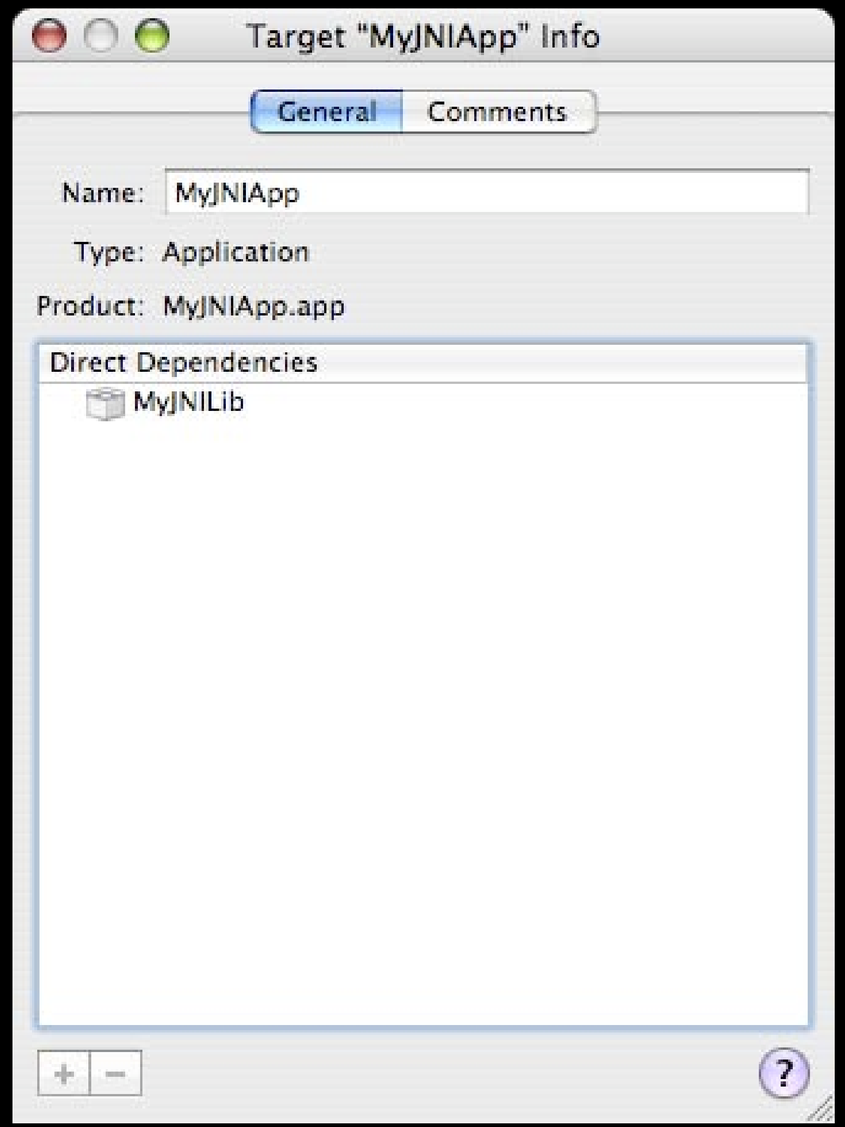
Native Library Target



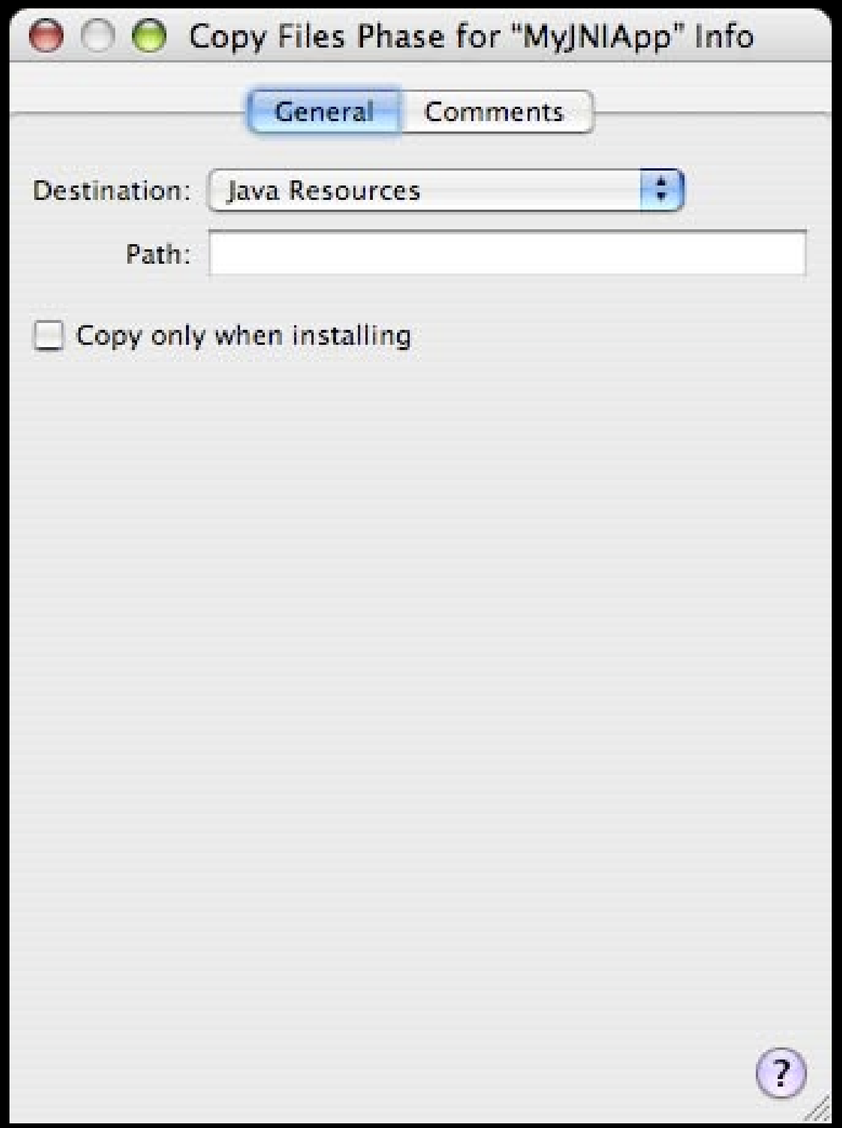
- All C/C++/ObjC code here
- Import generated headers or copy declarations
- Product name matches call to System.loadLibrary
- jnilib suffix (NOT dylib)
- lib prefix
- Include JavaVM.framework/Headers in search path

JNI Projects in Xcode

Target Dependencies



- Java Application target depends on Native Library target
- Headers target has no dependencies if source is included
- Native Library target depends on Headers target



JNI Projects in Xcode

Copy Files Build Phase

- Add Copy Files build phase to Java Application target
- Destination:“Java Resources”
- Drag JNI library over

Java-Native String Conversion

http://developer.apple.com/wwdc2005/

JNI Data Access: Strings

Creating a Java String from CFString/NSString

- Get 32-bit unicode characters from native string
  - `unichar` type interchangeable with `jchar`
- Create string with `NewString(jchar *)`
  - Do not call String constructor across JNI!
- UTF-8 strings also supported
  - `NewStringUTF`

```
unichar *cocoaBuf = [myNSString characters];
jobject javaStr = (*env)->NewString(env, (jchar *)cocoaBuf);
```

JNI Data Access: Strings

Creating a native string from a Java string

- Obtain a jchar* using `GetStringChars`
  - Interchangeable with unichar* type
- Do not directly manipulate Java characters!
  - Make a copy if modification is necessary
- Call `ReleaseStringChars` when finished

- Matching UTF-8 functions available
  - `GetStringUTFChars` returns `jbyte*` (char*)
  - `ReleaseStringUTFChars` also necessary

JNI Data Access: Strings

NSString from Java String

```
NSString *cocoaStr;
jchar *chars = (*env)->GetStringChars(env, jstr, NULL);
cocoaStr = [NSString stringWithCharacters:(unichar *)chars
   length:(*env)->GetStringLength(env, jstr)];
(*env)->ReleaseStringChars(env, message, chars);
```

Java-Cocoa GUI Integration

http://developer.apple.com/wwdc2005/

CocoaComponent

- Embed an NSView into AWT hierarchy
  - Extend `com.apple.eawt.CocoaComponent`
- Transmit or audit events right to Cocoa
- Low-cost integration

Extending CocoaComponent

- `createNSViewLong`
  - Implement as a native method
  - Create your NSView* and return as a (64-bit) `jlong`
  - 32-bit `createNSView` deprecated in Tiger
    - Call createNSViewLong and cast to int
- `sendMessage`
  - Convenience API for easy requests to native side
    - No need for `native` methods
  - Asynchronous call to `awtMessage:` Cocoa method
  - Easy for non-blocking requests to underlying NSView

QCCocoaComponent

- Uses Quartz Composer's QCView
- Display QC Compositions in a JFrame
- Extend Java class, implement native shell, move along

CocoaComponent Demo

http://developer.apple.com/wwdc2005/

JNI Data Access

Java Objects in C

- `jobject` datatype represents any java.lang.Object
  - `jstring`, `jclass` have their own types
- Store jobject references using `NewGlobalRef` or `NewWeakGlobalRef`
  - **GlobalRefs affect garbage collection!**
  - Use sparingly, remember to delete them
    - Use `removeNotify` for component peers

JNI Data Access

Calling Java methods from C

- Get method signature using `javap -s` in Terminal
- Always obtained from the class, not the object
  - Call `FindClass` or `GetObjectClass` to obtain a jclass reference
- Call appropriate JNI function using class and signature
  - `(*env)->GetMethodID(env, clazz, “myJavaMethod”, “()V”);`
  - also `GetStaticMethodID`

An Even Better CocoaComponent Demo!

http://developer.apple.com/wwdc2005/

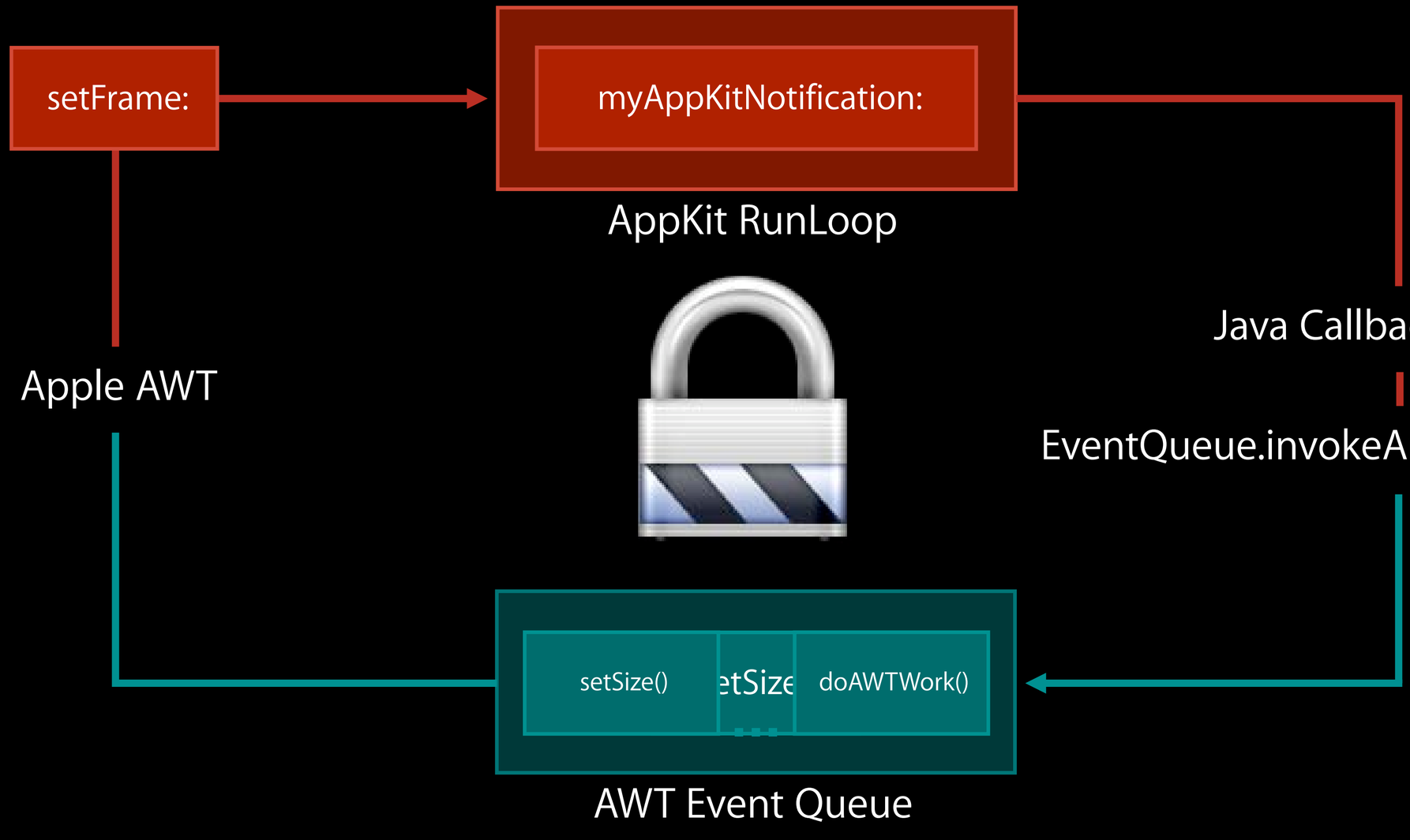
Threading Concerns

Java AWT vs Cocoa AppKit

- AWT and AppKit can coexist if you help them
- Two event queues on two different threads
- Do NOT block one with the other
  - Make all AWT-AppKit communication asynchronous
  - Use callbacks instead of return values

JNI Threading Concerns

Blocking AppKit-AWT Callbacks



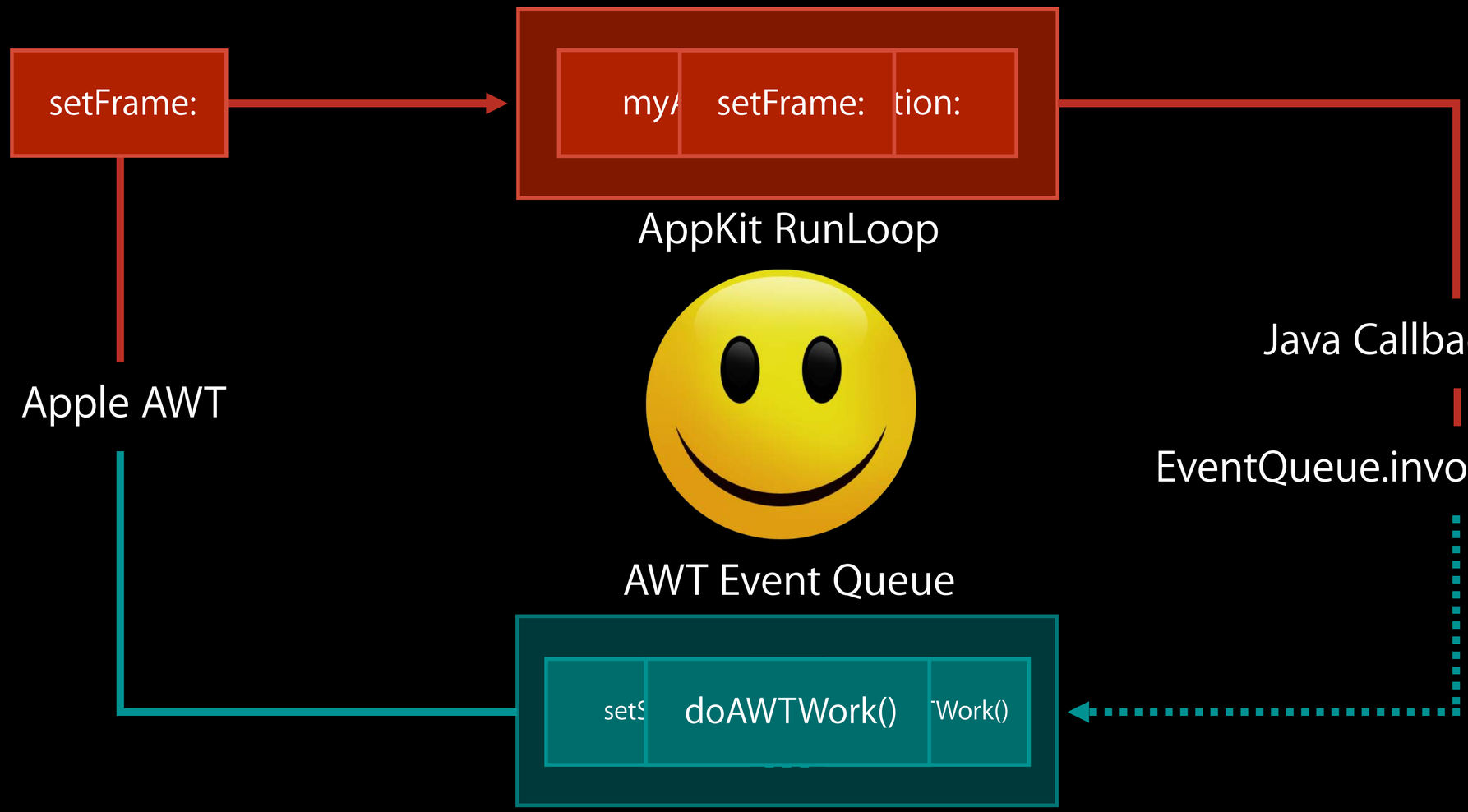
Java AWT vs. Cocoa AppKit

- Always call AWT using `EventQueue.invokeLater`
  - Same as `SwingUtilities.invokeLater`
  - NOT invokeAndWait

- Always call AppKit using `performSelectorOnMainThread:`
  - Pass NO for `waitUntilDone:`
  - CocoaComponent's `sendMessage` does this for you

JNI Threading Concerns

Asynchronous AppKit-AWT Callbacks



Threading Concerns

Watch Where You're Going!

- Be mindful of where you are
  - Use `Thread.currentThread`
  - Stay off "AWT-AppKit" in Java code
- Turn off CocoaComponent compatibility
  - `com.apple.eawt.CocoaComponent.CompatibilityMode=false`

AWT Native Interface (JAWT)

- Lower-level than CocoaComponent
- Draw into a Java canvas with native code
- Implement a native paint method
- Derive `MacOSXDrawingSurfaceInfo` struct
  - Platform-specific graphics/component contexts
- Get Component's NSView, and its parent window...

Demo
AWT Native Interface

http://developer.apple.com/wwdc2005/

Troubleshooting Deadlocks With gdb

- Native commands (output to gdb window):
  - bt
    - Prints backtrace of current native thread
  - t a a bt
    - Prints all threads

- Java commands (output to Java stdout):
  - call (void) ps()
    - Prints backtrace of current Java thread
  - call (void) pss()
    - Prints all threads

JNI Tips

- Store native pointers (NSView *) in Java as (64-bit) longs
  - Casting back safe on 32-bit systems
- Create a method/field ID cache at classload-time
  - jmethodID's/jFieldID's last as long as the class
  - No need to re-fetch every time
- Be careful with GlobalRefs
  - Have clear create/delete points
    - Use removeNotify for Components

Working With Mac OS X

- Use CFBundle APIs for locations within your app bundle
- Remember Objective-C Memory Management
  - retain, release, autorelease
    - CocoaComponent will retain your NSView for you
  - Create Autorelease Pools on new threads
    - createNSView, awtMessage have pools in place
- Throw Objective-C exceptions as Java exceptions
  - `(*env)->ThrowNew(env, excClass, “Uh-oh”);`

Starting a JVM From Cocoa

- Call JNI_CreateJavaVM
- Pass a JavaVMInitArgs struct
  - Request version (i.e. JNI_VERSION_1_4)
- Start the VM OFF the first thread!
  - NSThread detachNewThreadSelector:toTarget:withObject:
- Use JAVA_JVM_VERSION environment variable for J2SE 5.0
  - See updated samples
    - simpleJavaLauncher
    - JavaSplashScreen

Transitions

http://developer.apple.com/wwdc2005/

Transitions

What do Intel-based Macs mean for JNI applications?

- All JNI libraries must be universal
  - Start now!
- `JNI_CreateJavaVM` *will not work in Rosetta*
  - Build your native launchers as universal
- Endian-ness of constants, masks
  - Packed values will need to be converted

Related session

Java Virtual Machine Exposed

Session 647
Friday 10:30AM

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Who to Contact

Alan Samuel
Java Evangelist
bluker1@apple.com

Wiley Hodges
Java Product Manager
wiley@apple.com

Developer Technical Support
dts@apple.com

Related Sessions

Java Performance and Graphics Tuning	Session 632 Friday 9:00AM
Develop, Debug, Deploy Java on Mac OS X	Session 624 Thursday 9:00AM
Threading on Mac OS X	Session 124 Wed. 5:00PM
Maximizing Java Performance	Lab Friday 10:30AM

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

Alan Samuel
Java Evangelist
bluker1@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference